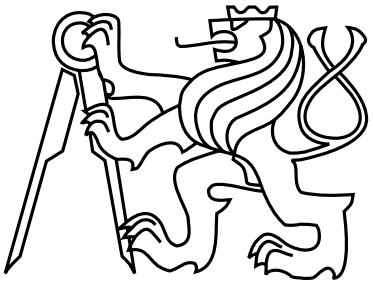
České vysoké učení technické v Praze
Fakulta stavební
Katedra geomatiky



Cvičení 4

Úloha č. 4: Množinové operace s polygony

Bc. Petr Poskočil a Bc. Marek Fáber

Vyučující: doc. Ing. Tomáš Bayer, Ph.D.

Studijní program: Geodézie a kartografie, Navazující magisterský

Obor: Geomatika

7. února 2020

Obsah

Kapitola 1

Zadání

Úloha č. 4: Množinové operace s polygony

Vstup: množina n polygonů $P=\{P_1, ..., P_n\}$.

Výstup: množina m polygonů $P' = \{P'_1, ..., P'_m\}$.

S využitím algoritmu pro množinové operace s polygony implementujte pro libovolné dva polygony $P_i, P_j \in P$ následující operace:

- Průnik polygonů $P_i \cap P_j$,
- Sjednocení polygonů $P_i \cup P_j$,
- Rozdíl polygonů: $P_i \cap \overline{P}_j$, resp. $P_j \cap \overline{P}_i$.

Jako vstupní data použijte existující kartografická data (např. konvertované shape fily) či syntetická data, která budou načítána z textového souboru ve Vámi zvoleném formátu.

Grafické rozhraní realizujte s využitím frameworku QT.

Při zpracování se snažte postihnout nejčastější singulární případy: společný vrchol, společná část segmentu, společný celý segment či více společných segmentů. Ošetřete situace, kdy výsledkem není 2D entita, ale 0D či 1D entita.

Pro výše uvedené účely je nutné mít řádně odladěny algoritmy z úlohy 1. Postup ošetření těchto případů diskutujte v technické zprávě, zamyslete se nad dalšími singularitami, které mohou nastat.

Hodnocení:

Krok	**Hodnocení**
Množinové operace: průnik, sjednocení, rozdíl	20b
Konstrukce offsetu (bufferu)	*+10b*
Výpočet průsečíků segmentů algoritmem Bentley & Ottman	*+8b*
Řešení pro polygony obsahující holes (otvory)	*+6b*
Max celkem:	**44b**

Obrázek 1.1: Zadání cvičení č. 4 [1]

1.1 Údaje o úlohách

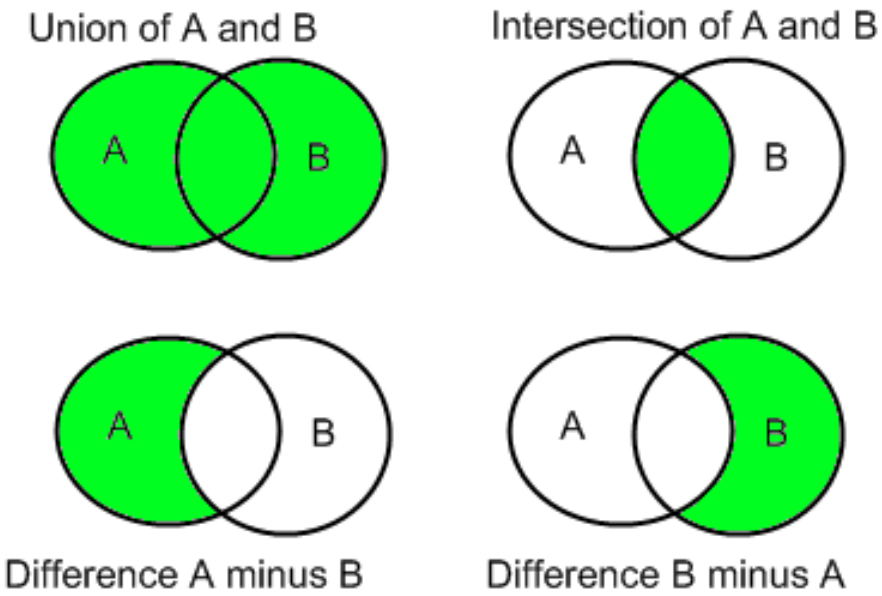
Povinnou částí úlohy je na množině dvou polygonů provést operace průniku, sjednocení a rozdílu. Z bonusové části nebyla zpracována žádná z úloh. [1]

Kapitola 2

Popis problému

Cílem úlohy je vytvořit aplikaci, do které bude možno nakreslit dva polygony A a B. Nad těmito polygony bude možné provádět Boolovské operace - sjednocení, průnik a rozdíl.

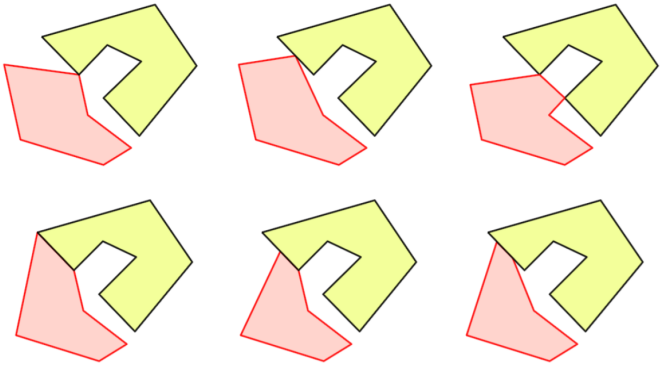
- *S*jednocení je operace, kdy výsledkem je množina, která obsahuje všechny části, které jsou obsaženy alespoň v jedné z množin.
- *P*růnik je operace, kdy výsledkem je množina, které obsahuje části, které jsou pro všechny množiny společné.
- *R*ozdíl je operace, kdy výsledkem je množina, která obsahuje části, které jsou součástí pouze jedné množiny.



Obrázek 2.1: Množinové operace [3]

U výše zmíněných matematických operací může docháze k takzvaným singulárním situacím, tj. situacím kdy výsledkem není pouze polygon, nýmrž i linie, či bod. Možné situace ve který se setkáme se simgularitami jsou vyzobrazeny na obrázku níže. Jmenovitě například:

- Společný vrchol,
- Vrchol ležící na hraně,
- Společná hrana,
- Či kombinace výše zmíněných



Obrázek 2.2: Singularity [2]

Kapitola 3

Popis použitých algoritmů

Pro tvorbu Booleovských operací je potřeba několika dílčí algoritmů popsaných níže v této kapitole. Nad naipotovanou, či graficky nakreslenou množino polygonů se provádí následující kroky.

- Průsečík dvou polygonů
- Vsunutí průsečíků do polygonů
- Ohodnocení vrcholů polygonů vůči sobě
- Výběr hran podle pozice
- Sestavení hran pro danou operaci

3.1 Výpočet průsečíků

Ve výpočtu dochází k určování vzájemného vztahu hran polygonů funkcí get2LinesPosition. Tato funkce určí, zda se hrany protínají, či nikoliv. Pokud se hrany protínají, funkce spočítá jejich průsečík a uloží ho do proměnné datového typu mapa. Do této promněnné se k výsledku ukládá také klíč, který odkazuje k dané hodnotě.

3.1.1 Implementace algoritmu

3.1.1.1 Výpočet průsečíků polygonů

1. $for(i = 0; i < n; i++)$
2. Vytvoření mapy: $M = map(double, QPointFB)$
3. $for(j = 0; j < m; j++)$
4. Existuje průsečík: $if(b_ij = (p_i, p_(i+)1\%n \cap q_j, q_(j+)1\%m)$
5. Přidej průsečík do M

6. Zpracuj první průsečík pro e_j : $ProcessIntersection(b_ij, \beta, B, j)$
7. Nějaké průsečíky jsme nalezli: $if(||M|| > 0)$
8. Procházej průsečíky v M: $for \forall m \in M$
9. Získej 2. hodnotu z páru: $b \leftarrow m.second$
10. Zpracuj první průsečík pro e_i: $ProcessIntersection(b, \alpha, A, i)$

3.1.1.2 Process Intersection

1. $if(t < \epsilon \wedge (t <= 1 - \epsilon)$
2. $i \Longleftarrow i + 1$
3. $P \Longleftarrow (begin + i, pi)$ Přidej průsečík na pozici 1

3.1.1.3 Poloha středového bodu hrany

1. for $(i = 0; i < n, i + +)$:
2. $M = \frac{p_i(x,y)+p_{i+1}(x,y)}{2}$ Výpočet středu hrany
3. $positionPointPolygonWinding(\bar{p}, B)$ Určení polohy M.
4. $p_i[pozice]$ = pozice

3.2 Množinové operace

Postup jednotlivých operací průnik, sjednocení, rozdíl je spojen do výsledného algoritmu.

3.2.1 Implementace algoritmu

3.2.1.1 selectEdges

1. $for(\forall P[i])$
2. if$(P[i]_{position} = position)$ Pozice bodu rovna hledané pozici
3. $e \leftarrow (P[i], P[i + 1])$ Vytvoř danou hranu
4. $edges \leftarrow e$

3.2.1.2 boolean operations

V této části se vybírají hrany pro zvolenou operaci. Pro každou operaci je nutno definovat vstupní typ. Pro operaci sjednocení se vybírají vnější hrany k polygonům A a B. Pro operaci průnik jsou naopak vybrány vnitřní hrany k polygonům A a B. Pro operaci rozdíl AB jsou vybrány vnější hrany k polygonu A, vniřní hrany z polygonu B a kolineární segmenty polygonu A. Pro operaci rozdíl BA, jsou to pak vniřní hrany z polygonu A, vnější hrany k polygonu B a kolineární segmenty polygonu B.

1. computePolygonIntersection(PA, PB) Nalejdi průsečíky polygonů
2. setPositionAB; Nastav hrany
3. if (Union)
4. seletEdges(PA, Outer)
5. selectEdges(PB, Outer) Sjednocení (Union)
6. if (Intersect)
7. seletEdges(PA, Inner)
8. selectEdges(PB, Inner) Průnik (Intersection)
9. if (DiffAB)
10. seletEdges(PA, Outer)
11. selectEdges(PB, Inner) Rozdíl AB (Difference AB)
12. selectEdges(PA, On) - singulární hrahy
13. if (DiffBA)
14. seletEdges(PA, Inner)
15. selectEdges(PB, Outer) Rozdíl BA (Difference BA)
16. selectEdges(PB, On) - singulární hrahy

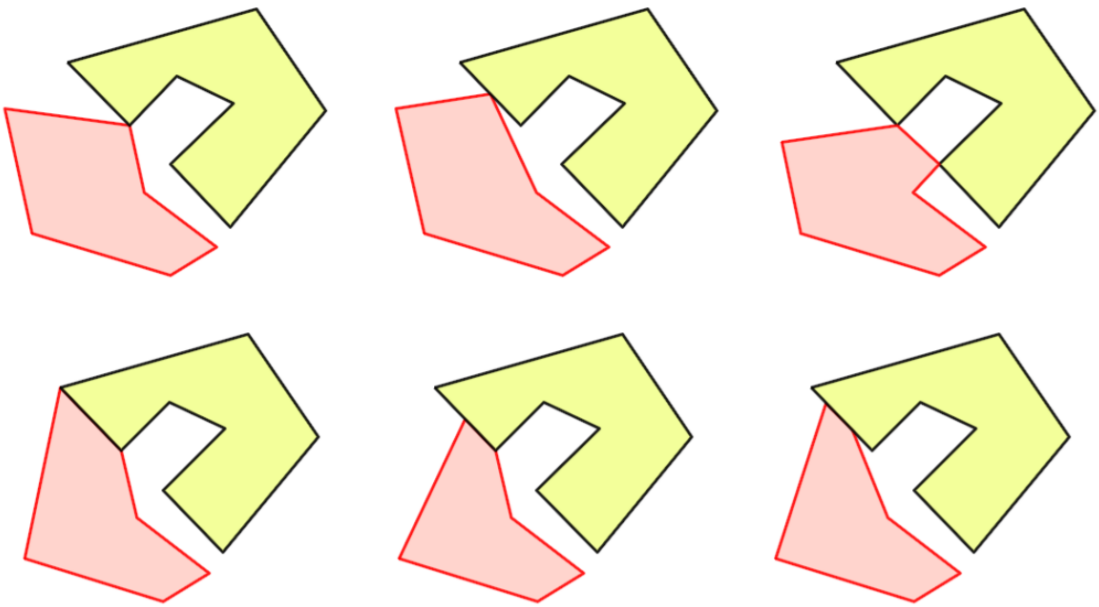
3.2.1.3 setPositionAB

1. Poloha bodů středů hran pro PA, PB
2. Poloha bodů středů hran pro PB, PA

3.3 Problematické situace

Mezi problematické situace patří například otvory v polygonech, jež v tomto algorimu nebyly řešeny. Tudíž u případného výskytu otvorů v polygonech by výsledná geometrie nebyla korektní.

Jelikož je k výpočtu použit Winding Number algorithms, je třeba počítat i se singularitami tohoto algoritmu. Můžou nastat situace, kdy polygony mají společný vrchol, hranu nebo vrchol jednoho polygonu leží na hraně polygonu druhého.



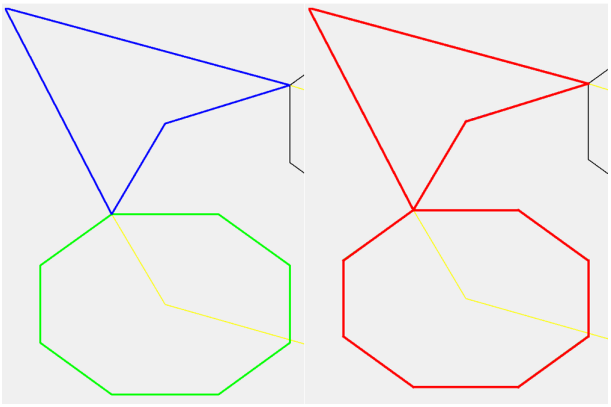
Obrázek 3.1: Singularity [2]

Pokud dojde k singulárnímu případu, kdy mají polygony společný bod, či společné hrany, tak pro požadavek Intersect je výsledkem prázdná množina hran. Pro požadavek Union u polygonů se společnou hranou je výsledkem polygon, který společnou hranu neobsahuje, jedná se tedy o celiství nově vzniklý polygon. Pro ostatní množinové operace je z podstaty jednotlivých operací výsledek klasický. Následný popis problému je generalizovaný na situaci společný bod a společná přímka. Možných prípadů je samozřejmě více, nicméně tyto dva přídady dostatečně popisují případy zbylé.

U situací, kde není možno rozlišit zda poloha bodu leží uvnitř nebo mimo polygon, tj. ležící na hraně (On), nebo pozici hrany která neleží uvnitř ani mimo, tj. taktéž (On), se jedná buďto o bod, nebo linii, resp. dimenzi entity menší než tři. Plocha možiny je tedy nulová a výsledkem operace průnik a sjednocení v místě dotyku je prázdná množina.

3.3.1 První singulární případ - společný bod

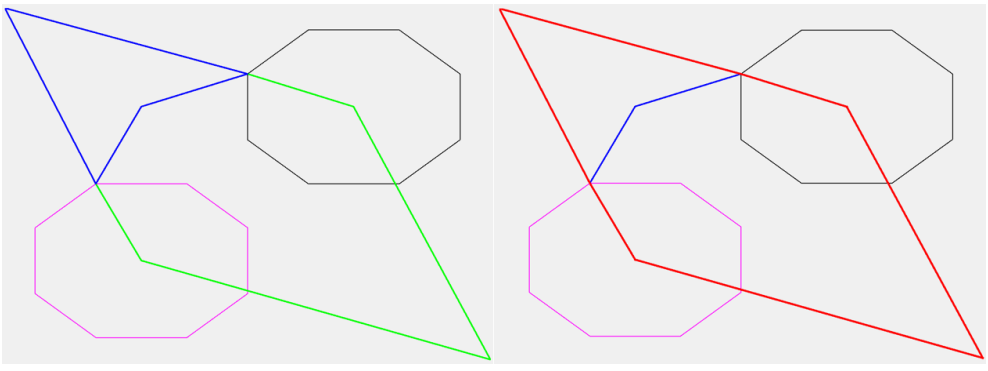
V této situaci nebyly u metody sjednocení ani u metody průnik uvažovány kolineární segmenty, neboď se nachází pouze v podobě jednoho nebo více bodů a tudíž je pro obě operace výsledkem prázdná množina v místě dotyku. Body samotné se nacháejí v množině, nicméně na hranici množiny, která v případě sjednocení v místě dotyku zaniká a vytváří tak jeden celistvý polygon. Výsledky operací znázorněny níže.



Obrázek 3.2: Singularity - Společný bod. Průnik (vlevo), Sjednocení (vpravo)

3.3.2 Druhý singulární případ - společná hrana

V této situaci též nebyli kolineární segmenty zahrnuty u metody sjednocení ani u metody průnik. Tato společná hrana se nachází na hanici množiny, nikoliv uvnitř. Z tohoto důvodu byly pro operaci sjednocení pouze vnější a naopak u operace průniku pouze vnitřní. Viz. obrázek níže. Operace rozdíl AB a rozdíl BA kolineární segment zahrnují, a to vždy segment polygonu určený prvním polygonem v rozdílu. Viz 3.2.1.2.



Obrázek 3.3: Singularity - Společná hrana. Průnik (vlevo), Sjednocení (vpravo)

Kapitola 4

Vstup a výstup aplikace

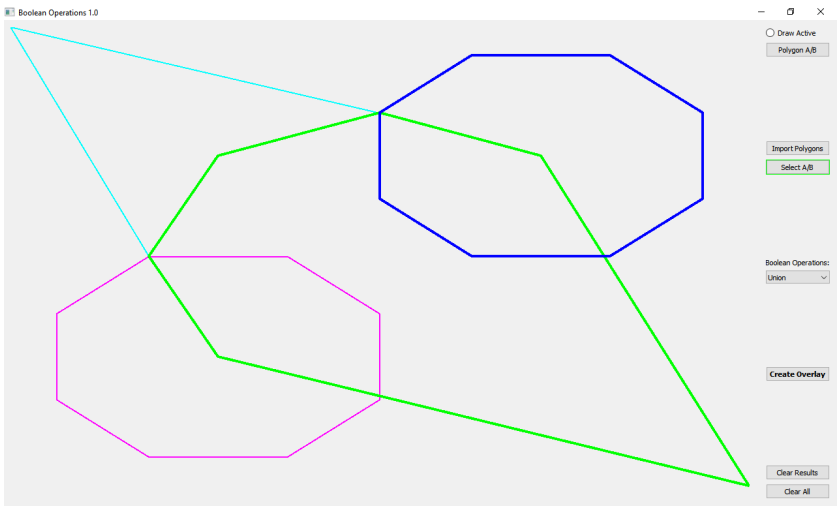
4.1 Vstup

Vstupními daty je množina polygonů, kterou uživatel může ručně nakreslit do grafické části aplikace nebo polygony do aplikace importovat z textového souboru.

4.2 Výstup

Výstupem úlohy je grafická aplikace, která nad vytvořenými polygony provádí množinové operace průniku, sjednocení a rozdílu. Do aplikace je možné importovat soubor s více polygony. Po vybrání dvou polygonů lze nad nimi provádět množinové analýzy.

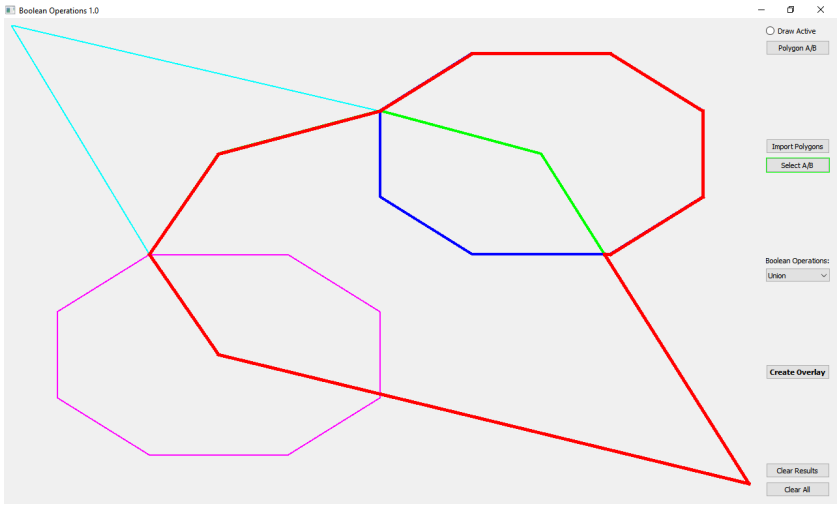
4.2.1 Aplikace



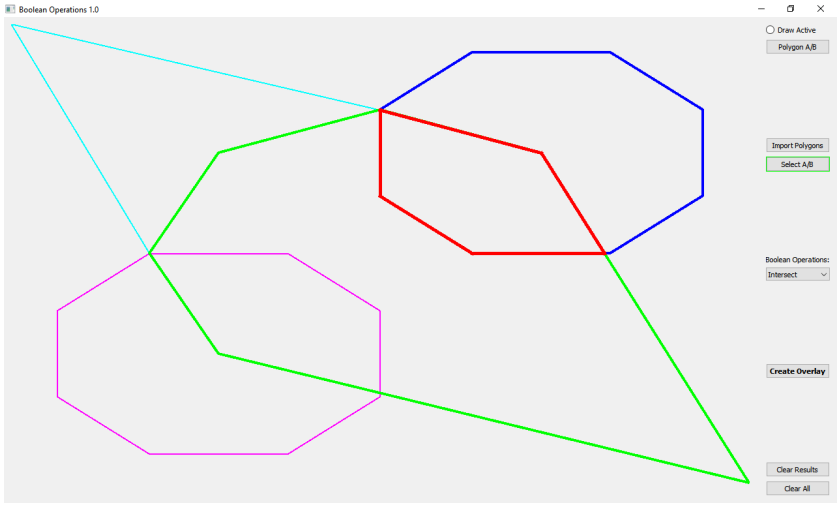
Obrázek 4.1: Ukázka aplikace s naimportovanými polygony

Pro analýzu byl vybrán zelený (A) a modrý (B) polygon. Výsledek je vždy zobrazen červenou barvou.

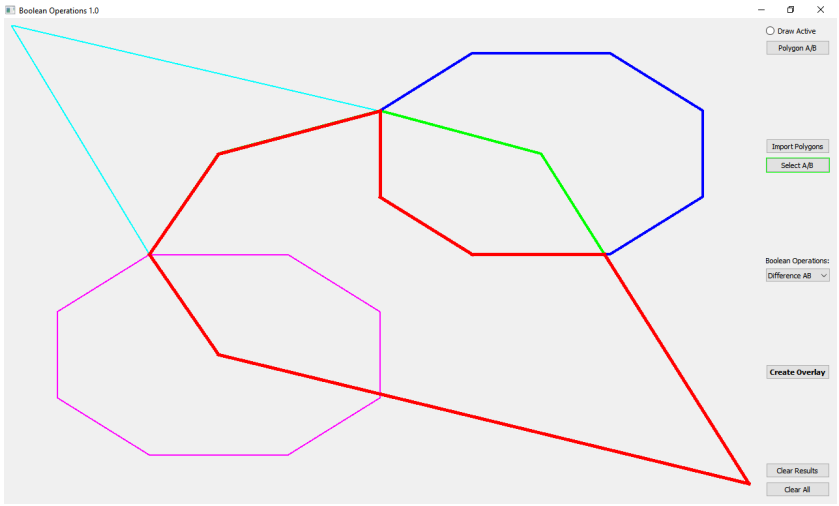
4.2.2 Výsledky analýz



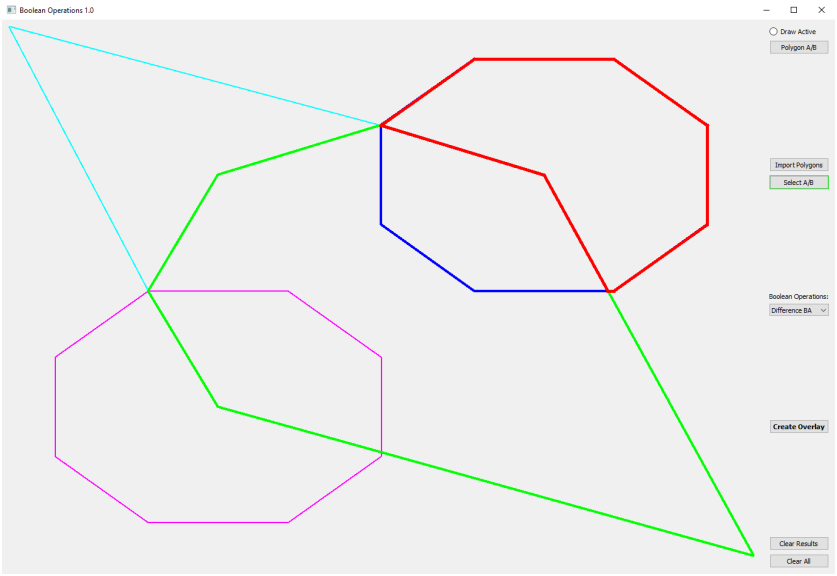
Obrázek 4.2: Sjednocení vybraných polygonů



Obrázek 4.3: Průnik vybraných polygonů



Obrázek 4.4: Rozdíl A-B vybraných polygonů



Obrázek 4.5: Rozdíl B-A vybraných polygonů

4.2.3 Analýza nad ručně nakreslenými polygony

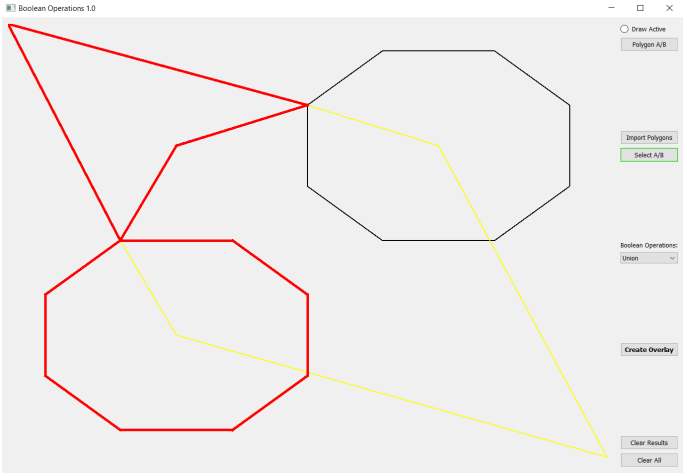
Obrázek 4.6: Sjednocení

Obrázek 4.7: Průnik

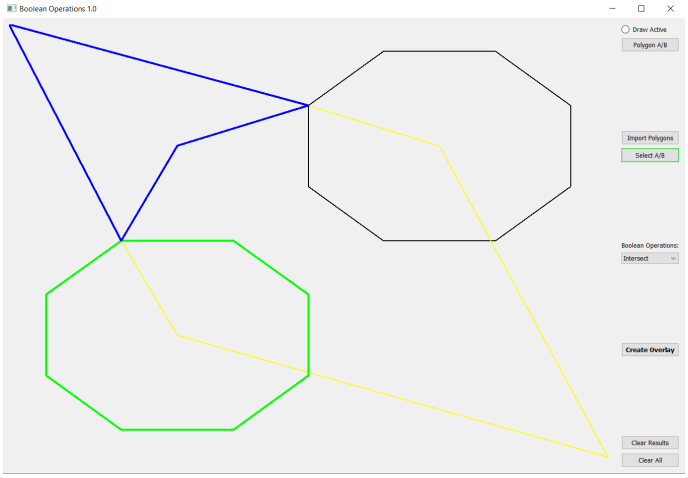
Obrázek 4.8: Rozdíl A-B

Obrázek 4.9: Rozdíl B-A

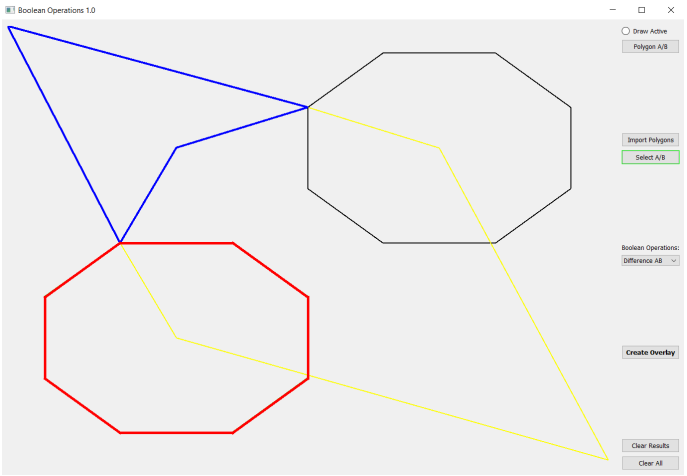
4.2.4 Singulární případy



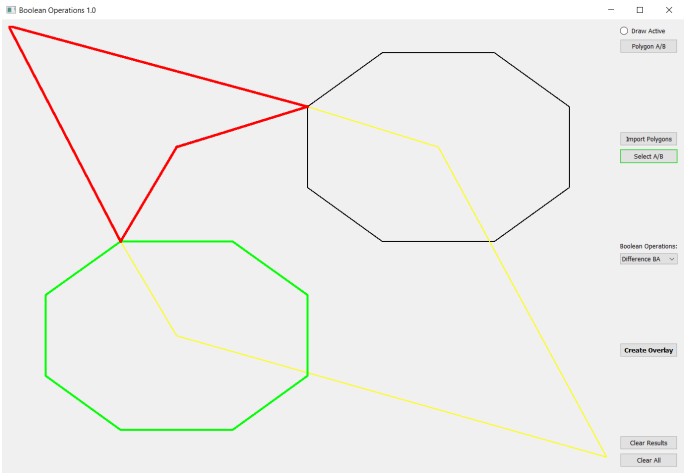
Obrázek 4.10: První Singulární případ - sjednocení



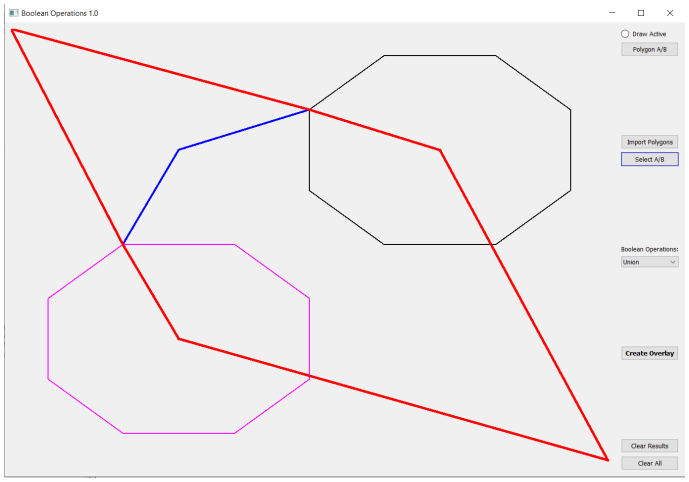
Obrázek 4.11: První Singulární případ - průnik



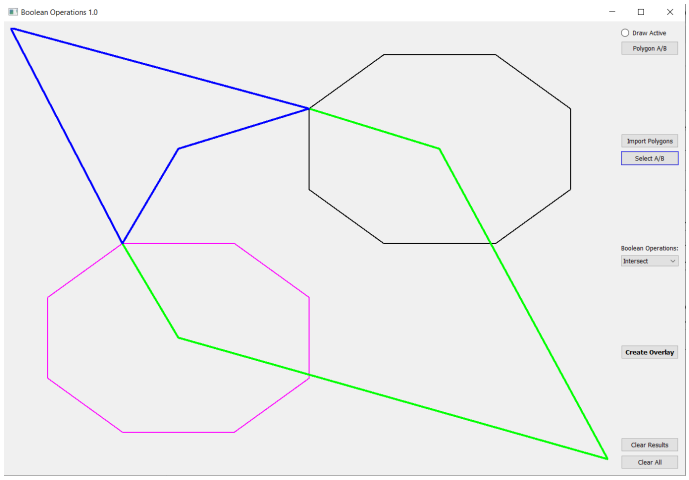
Obrázek 4.12: První Singulární případ - Rozdíl AB



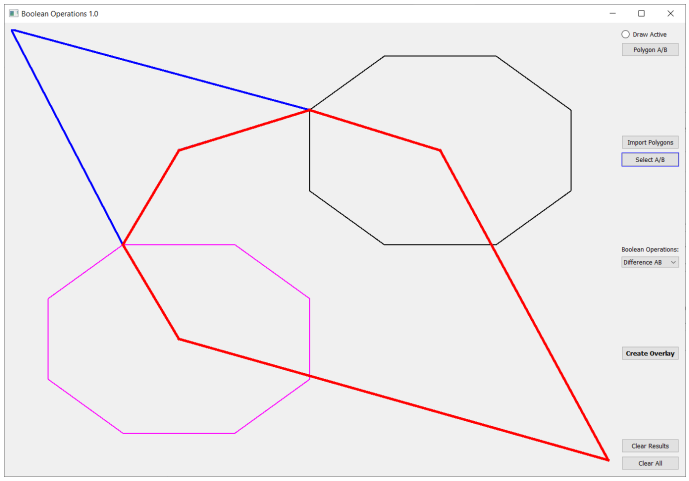
Obrázek 4.13: První Singulární případ - Rozdíl BA



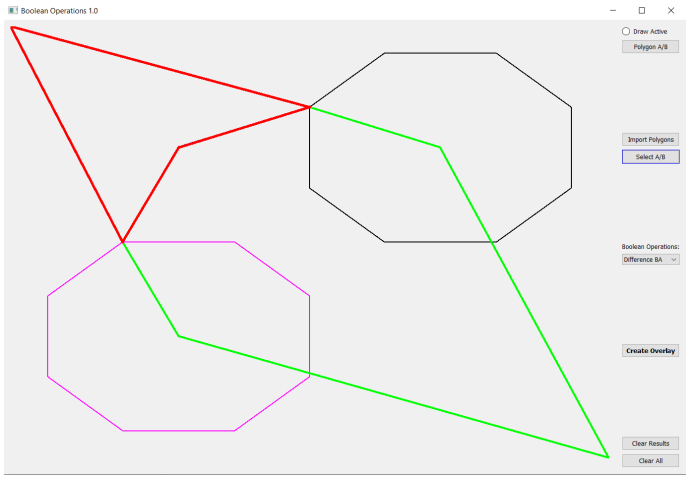
Obrázek 4.14: Druhý Singulární případ - sjednocení



Obrázek 4.15: Druhý Singulární případ - průnik



Obrázek 4.16: Druhý Singulární případ - Rozdíl AB



Obrázek 4.17: Druhý Singulární případ - Rozdíl BA

Kapitola 5

Dokumentace tříd a jejich metod

5.1 Algorithms

- *TPointLinePosition getPointLinePosition(QPointFB &q, QPointFB &p1, QPointFB &p2)*
  Tato funkce určuje pozici bodu vůči linii. Na vstupu funkce je určovaný bod a dva body přímky "a"a "b". Ze dvou bodů přímky můžeme určit determinant, jehož hodnotu porovnáváme se zvolenou minimální hodnotou "eps".

- *double getAngle2Vectors(QPointFB &p1, QPointFB &p2, QPointFB &p3, QPointFB &p4)*
  Tato funkce počítá úhel mezi dvěma hranami. Na vstupu jsou 4 body, které určují dvě přímky. Výstupem je velikost úhlu ve stupních typu double.

- *TPointPolygonPosition positionPointPolygonWinding(QPointFB &q, std::vector<QPointFB> &pol)*
  Tato metoda určí polohu bodu vůči polygonu.

- *T2LinesPosition get2LinesPosition(QPointFB &p1, QPointFB &p2, QPointFB &p3, QPointFB &p4, QPointFB &pi)*
  Metoda, která určí vztah dvou přímek a případně vypočítá jejich průsečík.

- *std::vector<Edge> booleanOperations(std::vector<QPointFB> &polygonA,std::vector<QPointFB> &polygonB, TBooleanOperation operation)*
  Metoda, která provádí Booleovské operace nad polygony.

- *void processIntersection(QPointFB &pi, double &t, std::vector<QPointFB> &polygon, int &i)*
  Metoda vloží bod b do polygonu na pozici i+1, pokud je na hraně daného polygonu a není vrcholem.

- *void computePolygonIntersection(std::vector<QPointFB> &pa, std::vector<QPointFB> &pb)*
  Metoda spočítá průsečíky polygonů a spustí *processIntersection*.

- *int getPositionWindingSelect(QPointF q, QPolygonF polygons)*
  Tato metoda určí polohu bodu vůči polygonu. Slouží k výběru polygonů při nahrání vlastních dat.

5.2 Draw

- *void changePolygon()*
  Metoda, která umožňuje přepínání mezi kresbou polygonu A a B.
- *void setA (std::vector<QPointFB> &a_)*
  Metoda pro nastavení polygonu A.
- *void setB (std::vector<QPointFB> &b_)*
  Metoda pro nastavení polygonu B.
- *std::vector<QPointFB> getA()*
  Metoda vracející polygon A.
- *std::vector<QPointFB> getB()*
  Metoda vracející polygon B.
- *void clearResults()*
  Metoda smaže výsledný polygon Boolovských operací.
- *void clearAll()*
  Metoda, která smaže grafické okno.
- *void mousePressEvent(QMouseEvent *event)*
  Metoda, která slouží ke vkládání bodů polygonu po kliknutí do grafického okna.
- *void paintEvent(QPaintEvent *event)*
  Metoda pro kreslení v grafickém okně.
- *void drawPolygon(QPainter &painter, std::vector<QPointFB> &polygon*
  Metoda, které vykresluje polygony.
- *static void importPolygons(std::string &path, std::vector<QPolygonF> &polygons, QSizeF &canvas_size*
  Metoda, která umožní importovat vlastní polygony.
- *void setRes (std::vector<Edge> res_)*
  Metoda pro nastavení výsledného polygonu Boolovských operací.

5.3 Edge

- *Edge(QPointFB &start, QPointFB &end)*
  Třída odvozená od třídy QPointF, která slouží k uložení bodu s výškou.

- *QPointFB &getStart()*
  Metoda, která vrátí počáteční bod hrany.
- *void setStart(QPointFB &s)*
  Metoda, která nastaví počáteční bod hrany.
- *QPointFB &getEnd()*
  Metoda, která vrátí koncový bod hrany.
- *void setEnd(QPointFB &e)*
  Metoda, která nastaví koncový bod hrany.

5.4 QPointfb

- *double getAlpha ()*
  Vrátí hodnotu alfa bodu třídy QPointFB.
- *double getBeta ()*
  Vrátí hodnotu beta bodu třídy QPointFB.
- *TPointPolygonPosition getPosition ()*
  Vrátí hodnotu position bodu třídy QPointFB.
- *void setAlpha (double alpha_)*
  Nastaví hodnotu alfa bodu třídy QPointFB.
- *void setBeta (double beta_)*
  Nastaví hodnotu beta bodu třídy QPointFB.
- *void setPosition (TPointPolygonPosition position_)*
  Nastaví hodnotu position bodu třídy QPointFB.

5.5 Widget

- *void on_ pushButton_ clicked*
  Umožní přepínat mezi kresbou polygonu A a B.
- *void on_ pushButton_ 2_ clicked*
  Spouští Booleovskou operaci.
- *void on_ pushButton_ 3_ clicked*
  Smaže výsledný polygon v grafické části.
- *void on_ pushButton_ 4_ clicked*
  Smaže polygony v grafické části.
- *void on_ pushButton_ 5_ clicked*
  Umožní načíst vlastní data do aplikace.

- *void on_ pushButton_ 6_ clicked*
  Umožňuje vybrat polygon A a B z vlastních dat.

- *void on_ radioButton_ clicked(bool checked)*
  Aktivuje vlastní kresbu polygonů.

Kapitola 6

Závěr

Ve vytvořené aplikaci lze vytvořit ručně nebo importovat množinu polygonů. Pomocí tlačítka "Select A/B"z importovaný polygonů je nutné vybrat dva polygony, mezi kterými bude docházet k výpočtu. O tuto funkci byl program rozšířen pro lepší demonstaci nad operacemi s rozmanitějším počtem polygonů.

6.1 Neřešené problémy a náměty

- Z časové tísně nebyly v úloze vyřešeny bonusové úlohy. Zejména by bylo vhodné řešit situaci, kdy polygony obsahují otvory.

Literatura

[1] T. Bayer. Množinové operace s polygony, cvičení č. 4 předmětu Algoritmy v digitální kartografii, 2016. https://web.natur.cuni.cz.

[2] T. Bayer. Množinové operace s polygony, přednáška č. 10 předmětu Algoritmy v digitální kartografii, 2016. https://web.natur.cuni.cz.

[3] J. Hughes. Množinové operace, 2020. https://people.stfx.ca/jhughes.

Příloha A

Zdrojový kód aplikace

Zdrojový kód aplikace je dostupný z veřejného profilu *petrposkocil* na github.com
Jméno repozitáře:*ADK_poskocil_faber*
Podsložka: *U4*

https://github.com/petrposkocil/ADK_poskocil_faber/U4/BooleanOperations

Obsažené soubory:
BooleanOperations.pro //Qt creator project file
algorithms.h //Header file
draw.h //Header file
edge.h //Header file
qpointfb.h //Header file
types.h //Header file
widget.h //Header file
main.cpp //Source code file
algorithms.cpp //Source code file
draw.cpp //Source code file
edge.cpp //Source code file
qpointfb.cpp //Source code file
widget.cpp //Source code file
widget.ui //User interface file

Příloha B

Aplikace - binární soubor

Aplikace je dostupná z veřejného profilu *petrposkocil* na github.com
Jméno repozitáře: *ADK_poskocil_faber*
Soubor: *BooleanOperations.exe*

https://github.com/petrposkocil/ADK_poskocil_faber/U4/BooleanOperations.exe

Příloha C

Výstupní data

Výstupní data jsou ve formě grafického znázornění množinových analýz dvou polygonů.